REMARKS/ARGUMENTS

Amendments were made to the specification to correct errors and to clarify the specification. No new matter has been added by any of the amendments to the specification.

Claims 1-20 are pending in the present application. No claims were added or canceled. Claims 1, 11, and 20 were amended. Support for amendments to the claims may be found in the Specification at least on page 22, lines 18-22, page 19, lines 3-24 and Figure 7. Reconsideration of the claims is respectfully requested.

I. Objection to the Specification

The Examiner has objected to the specification. The Examiner states:

> The disclosure is objected to because of the following informalities: The following application describes claims reference to a US Patent application that describes an exemplary mechanism for translating between access control list formats that is mentioned in the specification on page 9 paragraph number 2 but didn't provide the actual serial numbers. Appropriate correction is required.

Office Action dated December 14, 2006, page 2.

In response, the Specification has been amended to include the serial number of the US Patent Application that describes an exemplary mechanism for translating between access control list formats. Therefore, Applicants respectfully submit that the objections to the Specification have been overcome.

II. 35 U.S.C. § 101

The Examiner has rejected claim 20 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

The Examiner states:

> Claim 20 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claim 20 is directed to a computer readable medium. In this instance, this subject matter is not limited to that which falls within a statutory category of invention because it is not limited to a process, machine, manufacture, or a composition of matter. Instead, Applicant's specification provides intrinsic evidence on page 22 of the specification, it includes the operations and methods may be implemented in a software executable object or as a set of instructions embedded in a light wave and radio frequency.
> A set of instructions embedded in a light wave or a radio frequency are not limited to media which falls within a statutory category since they are clearly not limited to a mechanical device or combination of mechanical devices to constitute a machine, nor a tangible physical article or object which is some form of matter to be a product and constitute a manufacture, nor a composition of two or more substances to constitute a composition of matter.

> The Office's current position is that claims involving signals encoded with functional descriptive material do not fall within any of the categories of patentable subject matter set forth in 35 U.S.C. § 101, and such claims are therefore ineligible for patent protection. See 1300 OG 142 (November 22, 2005) (in particular, see Annex IV(c)).

Office Action dated December 14, 2006, pages 2-3.

This rejection is respectfully traversed. Claim 20 has been amended to recite "A computer program product, in a computer readable recordable-type medium, for managing access control lists in a filesystem...." Recording of a computer program on a computer readable recordable-type medium is a "manufacture" or "composition of matter" and, thus, within the statutory categories of invention. The computer readable medium recited in claim 20 does not include an intangible embodiment. Thus, claim 20 is directed towards statutory subject-matter. Therefore, the rejection of claim 20 under 35 U.S.C. § 101 has been overcome.

III. 35 U.S.C. § 102, Anticipation

The Examiner has rejected claims 1-3, 8, 10-13, 18 and 20 under 35 U.S.C. § 102(e) as being unpatentable by *Gai*, et al., Method and Apparatus for Organizing, Storing and Evaluating Access Control Lists, Patent No. 6,651,096, dated November 18, 2003 (hereinafter "*Gai*"). This rejection is respectfully traversed.

The Examiner states:

> **With the respect to claim 1 and 20**, Gai reference discloses:
> A method for managing access control lists in a filesystem (see abstract, "organizing, storing and evaluating access control lists"), the method comprising: associating two or more access control lists (see Fig. 4, elements 416a-416e) with a given filesystem object (see col. 7 lines 24-32; "The columns of the ACL represent the specific criteria with which network messages are compared"); responsive to receiving, from a requester, a request for an access control list associated with the given filesystem object (see col. 4, lines 26-32; col. 7, lines 24-32), determining a filesystem type of the requester (see col. 5, lines 13-21 ; col. 7, lines 29-34; col. 8, lines 9-15); and
> returning an access control list matching the filesystem type of the requester (see col. 8, lines 14-1 5, "Once a match is located, the corresponding action is returned and processing stops").
> **With the respect to claims 2 and 12**, Gai reference teaches determining whether an access control list matching the filesystem type of the requester exists (see col. 5, lines 13-21 ; col. 7, lines 29-34; col. 8, lines 9-1 5); and responsive to a determination that a matching access control list exists, returning the matching access control list (see col. 8, lines 14-15, "Once a match is located, the corresponding action is returned and processing stops") .
> **With the respect to claims 3 and 13**, Gai reference teaches wherein the step of returning the matching access control list (see col. 8, lines 14-15, "Once a match is located, the corresponding action is returned and processing stops") includes

accessing the matching access control list using an access mechanism (see col. 7, lines 24-27, ACE-Access Control Entry) associated with the filesystem type of the requester (see col. 5, lines 13-21 ; col. 7, lines 29-34; col. 8, lines 9-18). *Examiner notes the protocol field in the ACE is associated with the filesystem type of the requester and is used for access.*

With the respect to claims 8 and 18, Gai reference teaches wherein the step of associating two or more access control lists with a given filesystem object (see col. 7 lines 16-32) includes storing the two or more access control lists in file storage (see Fig. 4, element 408, NVRAM) with the given filesystem object (see col. 6, lines 1-2 & 13-1 8; col. 7, 60-66, "ACLs 416a-416e may be downloaded to device 316 ... and stored at NVRAM 408.").

With the respect to claim 10, Gai reference teaches wherein an access control list storage (see Fig. 4, element 410, TCAM) is provided an for each directory, each filesystem, or for each portion of a file system (see col. 6, lines 24-27, "apportioned segments 410a-e"; col. 6, lines 31 -35; col. 9, lines 22-31).

With the respect to claim 11, Gai reference discloses:

a filesystem (see abstract, "organizing, storing and evaluating access control lists"), wherein the filesystem includes a plurality of access mechanisms (see col. 7, lines 24-27, ACE- Access Control Entry) and wherein each access mechanism of the plurality of access mechanisms is associated with a filesystem type (see col. 5, lines 13-21 ; col. 8, lines 9-1 5); and a file storage (see Fig. 4, element 408), wherein the file storage has stored therein at least one filesystem object (see col. 7, lines 29-32) and wherein a given filesystem object within the at least one filesystem object has associated therewith two or more access control lists (see Fig. 4, elements 416a-e; col. 6, lines 15-18); wherein the filesystem, responsive to receiving from a requester a request for an access control list associated with the given filesystem object (see col. 4, lines 26-32; col. 7, lines 24-32), determines a filesystem type of the requester (see col. 5, lines 13- 21; col. 7, lines 29-34; col. 8, lines 9-1 5) and returns an access control list matching the filesystem type of the requester (see col. 8, lines 14-15, "Once a match is located, the corresponding action is returned and processing stops").

Office Action dated December 14, 2006, pages 3-6.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). *Gai* fails to teach each and every feature of independent claims 1, 11, and 20.

Independent claim 1 claims as follows:

A method for managing access control lists in a filesystem, the method comprising:

associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing filesystems;

responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor; and

returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor.

Independent claims 11 and 20 recite similar subject matter. *Gai* fails to teach the feature of a heterogeneous filesystem comprising two or more differing types of filesystems. *Gai* also fails to teach the steps for associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing filesystems; responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor; and returning an access control list for the given filesystem object matching the filesystem type of the requestor.

1. **Associating two or more access control lists with a given filesystem object**

Gai fails to teach "associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing filesystems," as is recited in amended claim 1. The Examiner cites to column 7, lines 24-32 of *Gai* which is included in the following section of *Gai*:

> First, a network administrator creates one or more access control lists in a conventional manner. For example, the administrator preferably utilizes a conventional text editor at a management station (not shown) to create the access control lists. FIGS. 5A-5E are highly schematic representations of text-based ACLs 416a-416e, respectively. Each access control list, such as ACL 416a, is given a name, such as ACL 101, and is preferably arranged in a table array having multiple rows and columns. Each row of the ACL, such as ACL 416a, corresponds to an Access Control Entry (ACE) statement, such as ACE statements 502-514, which specify the various criteria for the ACL 416a. The columns of the ACL represent the specific criteria with which network messages are compared. For example, ACLs 416a-416d each include a separate column for source address 516, destination address 518, source port 520, destination port 522 and protocol 524. Those skilled in the art will understand that greater or fewer message criteria may be employed. In addition, each ACL includes an action column 526 that corresponds to the particular action that is to be applied to network messages matching a corresponding ACE statement. In the preferred embodiment, permissible actions include permit, deny, permit and log, and deny and log.

Gai, column 7, lines 15-39.

Here, *Gai* teaches an access control list is arranged in a table array having multiple rows and columns. Each row in a single access control list corresponds to an access control entry (ACE) statement,

which specifies criteria for the access control list. In other words, *Gai* is describing the entries within a single access control list. *Gai* is not describing "associating two or more access control lists with a given filesystem object in a heterogenous filesystem." Moreover, the cited portion of *Gai* does not even mention a heterogenous filesystem, a filesystem object, or a filesystem that includes two or more differing types of filesystems. In fact, *Gai* does not even mention a filesystem or filesystem object in this or any other section of the reference.

The Examiner also cites to figure 4, elements 416a-416b of *Gai*. Figure 4 illustrates as follows:

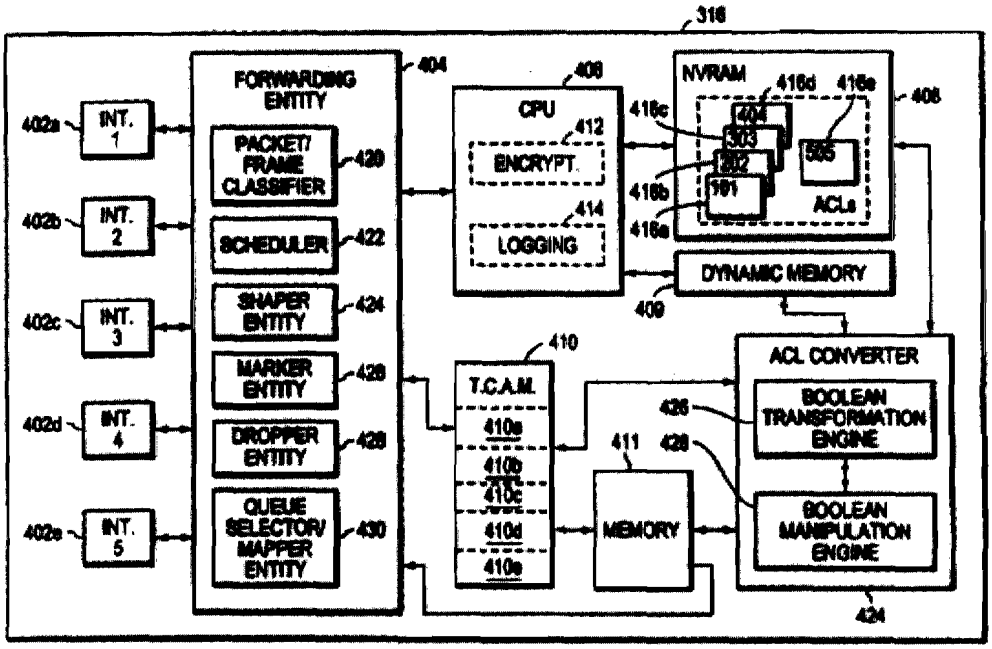

FIG. 4

As shown above, items 416a-416b in Figure 4 illustrates multiple access control lists in a non-volatile random access memory (NVRAM). The section of *Gai* describing Figure 4 states:

> CPU 406, which may be configured to run a plurality of executable functions, such as an encryption algorithm 412 and a logging function 414, is coupled to NVRAM 408, which may contain one or more text-based access control lists (ACLs) 416a-416e, such as ACLs 101, 202, 303, 404 and 505, and also to dynamic memory 409.

Gai, column 6, lines 13-18.

Thus, *Gai* illustrates storing multiple access control lists in NVRAM but *Gai* does not show multiple access control lists in a heterogeneous filesystem. Therefore, *Gai* fails to teach "associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing filesystems," as is claimed in claim 1.

2. Determining a filesystem type of the requestor

Gai fails to disclose "responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor," as is recited in claim 1. The Examiner cites to the following:

> With a single, unified ACL defined per interface per direction and stored in a CAM-type memory, the intermediate network device is able to rapidly evaluate network messages. In particular, upon receipt of a packet at a first interface, a forwarding entity at the intermediate network device tests the packet against the single, unified ACL stored in the corresponding portion of the CAM. When a match is obtained, the corresponding decision is returned to the forwarding entity, which then takes the appropriate action (e.g., forward, discard, log and forward, transfer to CPU for additional processing, etc.) Since the intermediate network device only evaluates a single, unified ACL stored in the CAM, rather than multiple ACLs stored in RAM, a decision can be rapidly obtained.

Gai, column 4, lines 26-39.

This portion of *Gai* states that a single, unified access control list for an interface can be used to evaluate network messages. Thus, *Gai* relies on merging multiple access control lists into a single, unified access control list rather than associating two or more access control lists with a filesystem object and determining a filesystem type of the requestor to return an access control list from the two or more access control lists for the filesystem object that matches the filesystem type of the requestor, as in claim 1. *Gai* does not determine a filesystem type of a requestor in response to receiving a request for an access control list associated with a given filesystem object. Moreover, even if the unified access control list of *Gai* could be equivalent to the unified access control list, *Gai* does not teach a heterogeneous filesystem. Therefore, it would be completely unnecessary for *Gai* to determine a filesystem type of a requestor.

The Examiner also cites to the following:

> FIG. 3 is a highly schematic block diagram of a computer network 300. Network 300 includes a plurality of local area networks (LANs), such as LAN 302 and 304, each of which may be associated with a different department, such as human resources and engineering, respectively. A plurality of end stations, such as end stations 306-312, and servers, such as servers 313 and 314, may be coupled to LANs 302, 304. LANs 302 and 304 may also be interconnected by an intermediate network device 316. Device 316 may also provide LANs 302 and 304 with connectivity to other networks, such as the well-known Internet 318. Software entities (not shown) executing on the various end stations 306-312 and servers 313 and 314 typically communicate with each other by exchanging discrete packets or frames according to predefined protocols, such as the Transmission Control Protocol/Internet Protocol (TCP/IP), the Internet Packet Exchange (IPX)

> protocol, the AppleTalk protocol, the DECNet protocol or NetBIOS Extended User Interface (NetBEUI).

Gai, column 3, lines 3-21.

Here, *Gai* generally describes network protocols for defining how entities interact with each other, such as the transmission control protocol/Internet protocol (TCP/IP) protocol for transmitting data packets over the Internet. However, *Gai* does not teach determining a filesystem type of a requestor in this or any other section of the reference. The Examiner also cites to *Gai* at column 7, lines 24-32 and lines 29-34, which is quoted above. As discussed above, *Gai* describes access control entries in an access control list. *Gai* does not disclose determining a filesystem type of a requestor in this or any other section of the reference. Furthermore, any teachings regarding access control list entry statements cannot teach determining a filesystem type of the requestor requesting an access control list associated with a given filesystem object. Thus, *Gai* fails to teach "responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor," as is claimed in amended claim 1.

3. Returning an access control list for the given filesystem object

Gai also fails to teach "returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor," as is claimed in claim 1. *Gai* states:

> For example, the network administrator may assign ACL 416a (ACL 101) to interface 402a for purposes of input security control. Accordingly, upon receipt of a network message at interface 402a, it is compared with ACE statements 502-514 of ACL 416a. The matching is preferably performed logically as a series of sequential steps starting with the first ACE and moving, one ACE at a time, toward the last ACE in the ACL. Once a match is located, the corresponding action is returned and the processing stops. That is, no additional ACEs are examined. If a match is made with an ACE statement having a "permit" action (e.g., ACE 502), the packet is forwarded. If a match is made with an ACE statement having a "deny" action (e.g., ACE 506), the packet is dropped. If the matching action is "permit and log", then the respective message is forwarded and an entry is made in a conventional message log. Similarly, if the matching action is "deny and log", then the respective message is dropped and a log entry made. If no ACE of the subject ACL matches the message, an implicit action located at the end of the ACL is typically returned (e.g., permit or deny).

Gai, column 8, lines 7-27.

This cited portion of *Gai* discusses comparing a network message with access control entry statements of an access control list. If a match is found, an action corresponding to the action in the access control entry is taken. As discussed above, *Gai* does not teach or even mention a filesystem or filesystem object. Moreover, comparing access control list entries in a single access control list with a

network message and performing corresponding actions if a match is found cannot be construed to disclose returning an access control list from two or more access control lists matching the filesystem type of the requestor. Therefore, *Gai* fails to teach "returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor," as is recited in claim 1.

Thus, independent claim 1 is not anticipated by *Gai* because *Gai* fails to teach each and every feature of claim 1. In addition, independent claims 11 and 20 recite subject matter discussed above with regard to claim 1. Therefore, claims 11 and 20 are distinguishable over *Gai* at least for the reasons set forth above with regard to claim 1.

Dependent claims 2-3, 8, 10, 13, and 18 depend from claims 1 and 11. Therefore, claims 2-3, 8, 10, 13, and 18 are distinguishable over *Gai* at least by virtue of their dependency on claims 1 and 11. In addition, claims 2-3, 8, 10, 13, and 18 recite additional combinations of features that are not disclosed by *Gai.* For example, regarding claims 3 and 13, *Gai* fails to teach returning the matching access control list includes accessing the matching access control list using an access mechanism associated with the filesystem type of the requestor. The Examiner alleges this feature is disclosed by *Gai* at column 7, lines 24-27 and lines 29-34, which is quoted above. As discussed above, this section of *Gai* discloses access control entry statements in an access control list. Such teachings do not disclose access the matching access control list using an access mechanism associated with the filesystem type of the requestor. In fact, a filesystem type of the requestor, or any other user, is not mentioned anywhere in this or any other section of the reference.

The Examiner also cites to column 5, lines 13-21, which is also quoted above. As discussed above, this portion of *Gai* generally discusses protocols, such as the transmission control protocol/Internet protocol (TCP/IP) protocol for communicating data packets over a network. Such general teachings regarding network protocols cannot be construed to disclose anything regarding a filesystem, a filesystem objects, or an access mechanism associated with the filesystem type of the requestor.

The Examiner also cites to *Gai* at column 8, lines 9-18 which is also quoted above. As discussed above, this portion of *Gai* describes comparing a network message with access control entry statements in an access control list. Although *Gai* discusses matching access control entry statements, such matching cannot be construed to explicitly or implicitly teach access a matching access control list **using an access mechanism** associated with the **filesystem type of the requestor**. Therefore, *Gai* fails to teach each and every feature of claims 3 and 13. Therefore, the rejection of claims 1-3, 8, 10-13, 18 and 20 under 35 U.S.C. § 102(e) has been overcome.

Furthermore, *Gai* does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. *Gai* actually teaches away from the presently claimed invention

because it teaches generating a single, unified access control list for a given interface as opposed to returning an access control list from the tow or more access control lists for a given filesystem object matching the filesystem type of a requestor, as in the presently claimed invention. Absent the examiner pointing out some teaching or incentive to implement *Gai* and returning an access control list matching the filesystem type of the requestor, one of ordinary skill in the art would not be led to modify *Gai* to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify *Gai* in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

IV. 35 U.S.C. § 103, Obviousness

The Examiner has rejected claims 4-7, 9, 14-17 and 19 under 35 U.S.C. § 103(a) as being unpatentable over *Gai* in view of *Hitz* et al., File Access Control in a Multi-Protocol File Server, Patent No. 6,457,130 dated September 24, 2002 (hereinafter "*Hitz*"). This rejection is respectfully traversed.

The Examiner states:

> **With the respect to claims 4 and 14,** Gai reference teaches further comprising: responsive to a determination that a matching access control list does not exist (see col. 8; lines 24-26; col. 7, lines 24-27, "If no ACE of the subject ACL matches the message, an implicit action located at the end of the ACL is typically returned),
> Gai reference doesn't teach providing a new access control list for the filesystem type of the requestor; and returning the new access control list.
> Hitz reference teaches responsive to a determination that a matching access control list does not exist (see col. 6, lines 1-2), providing a new access control list for the filesystem type of the requester (see col. 8, lines 26-34, new access control limits); and returning the new access control list (see col. 8, lines 12-16; col. 8, lines 35-40; col. 8, lines 60-62).
> It would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains to have Gai invention and included the step of providing a new access control list for the filesystem type when a matching access control list does not exist of the requester and returning the new ACL for the purpose of enforcing file access control among client devices using multiple diverse access control models and multiple diverse file server protocols (see Hitz col. 2, lines 36-40).
> **With the respect to claims 5 and 15,** Hitz reference teaches wherein the step of returning the new access control list (see col. 8, lines 12-16; col. 8, lines 12-16; col. 8, lines 35-40; col. 8, lines 60-62) includes accessing the new access control list (see col. 8, lines 26-29; "When the file has its access control limits modified") using an access mechanism associated with the filesystem type of the requester (see col. 4, lines 8-1 1 & lines 43-56, ACE-access control entries).
> **With the respect to claims 6 and 16,** Hitz reference teaches wherein the step of

providing a new access control list for the filesystem type of the requestor (see col. 8, lines 26-34, new access control limits) includes translating an existing access control list to the filesystem type of the requester (see col. 6, lines 1-1 0). **With the respect to claims 7 and 17**, Hitz reference teaches wherein the step of providing a new access control list for the filesystem type of the requester (see col. 8, lines 26-34, new access control limits) includes providing a default access control list for the filesystem type of the requester based on rules associated with the filesystem (see col. 6, lines 10-1 3).
With the respect to claims 9 and 19, Gai reference teaches wherein the step of associating two or more access control lists with a given filesystem object (see Gai col. 7 lines 16-32). Gai reference doesn't teach storing a native access control list in file storage with the given filesystem object and storing one or more non-native access control lists in access control list storage separate from the file storage. Hitz reference teaches storing a native access control list (see col. 4, lines 8-11, "NT ACL") in file storage (see Fig. 1, element 112; col. 4, lines 43-48, NT security style) with the given filesystem object and storing one non-native access control list (see col. 4, lines 8-1 1, "Unix Perms") in access control list storage separate from the file storage (see col. 4, lines 8-25, Unix security style). It would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains to have the native and non-native access control list stored separately for the purpose of enforcing file access control among client devices using multiple diverse access control models and multiple diverse file server protocols (see Hitz col. 2, lines 36-40).

Office Action dated December 14, 2006, pages 7-9

As discussed above, *Gai* fails to teach or suggest the feature of a heterogeneous filesystem and associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing filesystems; responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor; and returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor. Moreover, *Hitz* fails to make up for the deficiencies of *Gai*. *Hitz* is directed toward providing a method and system for enforcing file access control among client devices using multiple diverse file server protocols. *Hitz* teaches:

> The invention provides a method and system for enforcing file access control among client devices using multiple diverse access control models and multiple diverse file server protocols. A multi-protocol file server identifies each file with one particular access control model out of a plurality of possible models, and enforces that one particular model for all accesses to that file. When the file server receives a file server request for that file using a different access control model, the file server translates the access control limits for that file into no-less-restrictive limits in the different model. The file server restricts access by the client device using the translated access control limits. Each file is assigned the access control model of the user who created the file or who last

set access control limits for the file. When a user having a different access control model sets access control limits, the access control model for the file is changed to the new model. Files are organized in a tree hierarchy, in which each tree is limited to one or more access control models (which can limit the ability of users to set access control limits for files in that tree). Each tree can be limited to NT-model-only format, Unix-model-only format, or mixed NT-or-Unix-models format.

Hitz, abstract.

Here, *Hitz* describes a file server that receives a file server request and translates the access control limits for that file into no-less-restrictive limits in a different access control model. However, *Hitz* does not teach or suggest a heterogeneous file system including two or more different types of file systems. In fact, *Hitz* teaches a single file system as shown in Figure 1 of *Hitz* which illustrates as follows:

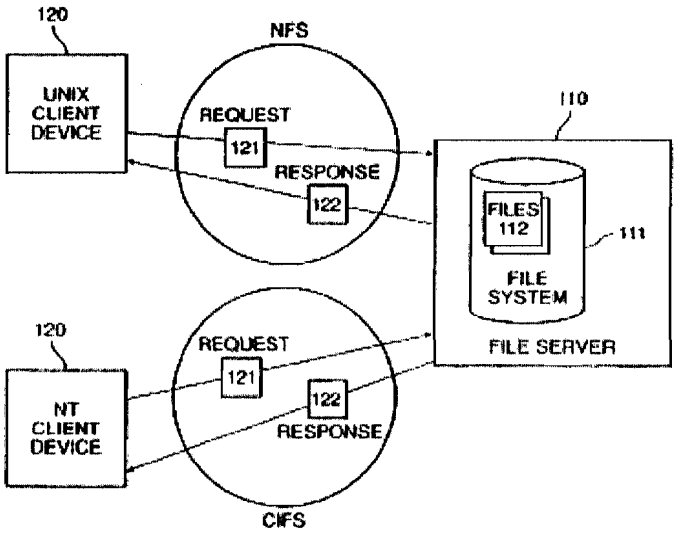

FIG. 1

Hitz, Figure 1.

Here, *Hitz* illustrates a single filesystem receiving requests from clients. *Hitz* does not teach or suggest a heterogeneous filesystem having two or more different types of filesystem and determining a filesystem type of the requestor, as claimed in claim 1. Moreover, the section of *Hitz* describing Figure 1 states:

> The figure shows a block diagram of a system for enforcing diverse access control models among client devices.
>
> A system 100 includes a file server 10, and a set of client devices 120.
> The file server 110 maintains a file system 111, including a set of files 112.

Hitz, column 3, lines 21-26.

Thus, *Hitz* fails to teach or suggest a problem involving multiple different types of filesystems. Moreover, *Hitz* does not teach or suggest returning an access control list from the two or more access control lists for the filesystem object matching the filesystem type of the requestor. Therefore, claims 4-7, 9, 14-17 and 19 are not obvious in view of *Hitz* because the features believed to be disclosed by this cited reference are not present.

Moreover, the present invention recognizes the problem of managing access control lists in a heterogeneous filesystem. *Hitz* and *Gai* do not teach this problem. *Gai* is directed to the problem of multiple access control lists assigned to a single interface. *Gai* teaches

> Access control lists are primarily used to provide security. Thus, for a given interface, only a single list is evaluated per direction. For purposes of security, moreover, the lists are relatively short. Nevertheless, the evaluation of such lists by software modules can significantly degrade the intermediate device's performance (e.g., number of packets processed per second). This degradation in performance has been accepted mainly due to a lack of acceptable alternatives. It is proposed, however, to expand the use of access control lists for additional features besides just security decisions. For example, access control lists may also be used to determine whether a given packet should be encrypted and/or whether a particular quality of service (QoS) treatment should be applied. Accordingly, it is anticipated that multiple access control lists may be assigned to a single interface. As additional access control lists are defined and evaluated per packet, the reduction in performance will likely reach unacceptable levels. Accordingly, a need has arisen to optimize the creation and evaluation of multiple access control lists so as to maintain, if not improve, packet processing speeds. This is especially true as more and more internetworking functionality is being implemented in hardware circuitry to increase the speed and performance of internetworking devices.

Gai, column 3, lines 17-40.

As shown above, *Gai* is directed to solving the problem of slow data packet processing speeds due to multiple access control lists assigned to a single interface. Thus, *Gai* fails to teach the problem disclosed by the present invention.

Hitz is directed to the problem of enforcing file security among multiple diverse access control models and file server protocols in a file system intended for use by more than one user. *Hitz* teaches:

> Accordingly, it would be desirable to provide a method and system for enforcing file security semantics among client devices using multiple diverse access control models and multiple diverse file server protocols. This advantage is achieved in an embodiment of the invention in which a multi-protocol file server identifies each file with one particular access control model out of a plurality of possible access control models, and enforces that particular access control model for all accesses to that file. When the file server receives a file server request for that file using a file server protocol with a different access control model, the file server translates the access control limits imposed by the file's access control model into no-less-restrictive access control limits in the different

> access control model. The file server restricts access to the file using the translated access control limits.

Hitz, column 2, lines 19-34.

Thus, *Hitz* does not teach the problem of managing access control lists in a heterogeneous filesystem.

In addition, *Hitz* and *Gai* do not teach the solution disclosed in the claimed invention. *Gai* solves the problem of multiple access control lists assigned to a single interface by generating a unified access control list. *Gai* states as follows:

> Briefly, the invention relates to a method and apparatus for efficiently organizing, storing and evaluating access control lists ("ACLs"). According to the invention, an ACL converter comprises a boolean transformation engine cooperatively coupled to a boolean manipulation engine for optimizing one or more text-based ACLs for subsequent evaluation by an intermediate network device. The boolean transformation engine accesses the one or more ACLs and translates them into a first boolean representation. In the preferred embodiment, the first boolean representation is a binary decision diagram (BDD). The boolean manipulation engine then optimizes and merges the ACLs specified for a given interface of the device. That is, the boolean manipulation engine performs one or more operations on the specified ACLs (in BDD format) to generate a single, unified ACL for the given interface. In order to prioritize the possibly conflicting actions output by the ACLs assigned to a given network message, the ACL converter preferably utilizes one or more predefined conflict resolution tables during the merging process.

Gai, column 3, lines 55-column 4, lines 8.

As shown above, *Gai* discloses a single, unified access control list for a given interface. *Gai* does not teach or suggest returning an access control list from two or more access control lists for a given filesystem object matching the filesystem type of a requestor, as in claim 1. Thus, *Gai* fails to teach the problem or the source.

Hitz also fails to teach the solution disclosed by the present invention. *Hitz* solves the problem of enforcing file security among multiple diverse access control models and file server protocols by identifying a particular access control model for a file server and enforcing that model for all accesses to that file. *See* abstract. *Hitz* does not determine a filesystem type of a requestor and return an access control list from two or more access control lists for a filesystem object matching the filesystem type of the requestor. Therefore, *Hitz* and *Gai* fail to teach the problem and the solution disclosed in the present invention in claim 1.

Moreover, the examiner may not use the claimed invention as an "instruction manual" or "template" to piece together the teachings of the prior art so that the invention is rendered obvious. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Such reliance is an impermissible use of hindsight with the benefit of applicant's disclosure. *Id.* Therefore, absent some teaching, suggestion, or incentive in the prior art, *Gai* and *Hitz* cannot be properly combined to form the claimed invention. As a

result, absent any teaching, suggestion, or incentive from the prior art to make the proposed combination, the presently claimed invention can be reached only through an impermissible use of hindsight with the benefit of applicant's disclosure a model for the needed changes. Therefore, the rejection of claims 4-7, 9, 14-17 and 19 under 35 U.S.C. § 103(a) has been overcome.

V. Conclusion

It is respectfully urged that the subject application is patentable over *Gai* in view of *Hitz* and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: March 14, 2007

Respectfully submitted,

/Mari Stewart/

Mari Stewart
Reg. No. 50,359
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants